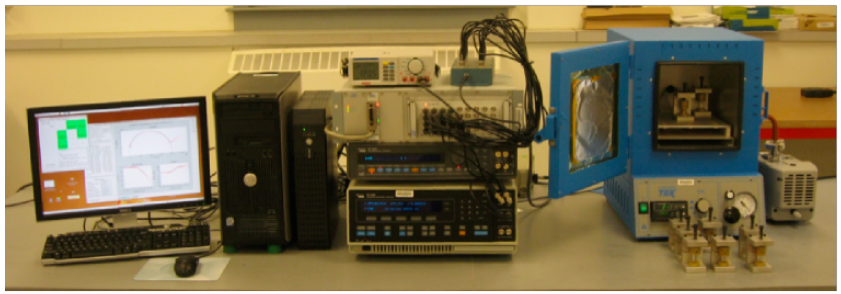
Using Open-Source Scripting Languages for Rapid-Development of Informatics Capabilities

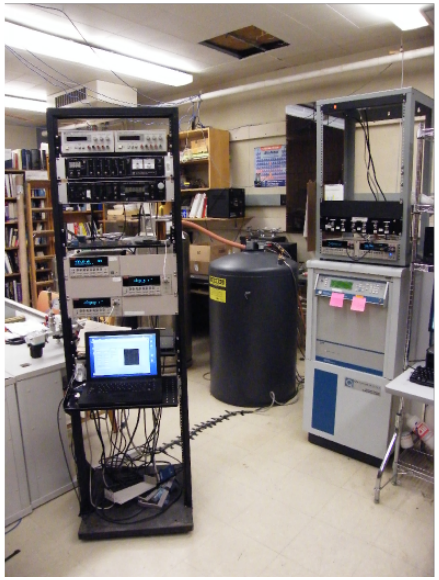
Craig Versek, Michael Thorn, Mark Tuominen

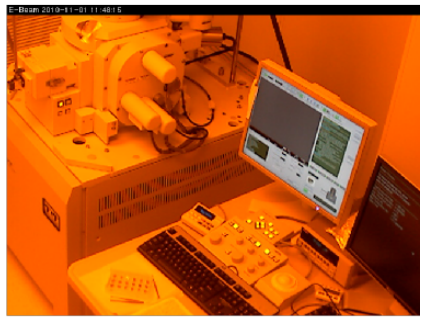
Department of Physics
UMass Amherst

November 4th, 2010

Lab Objectives





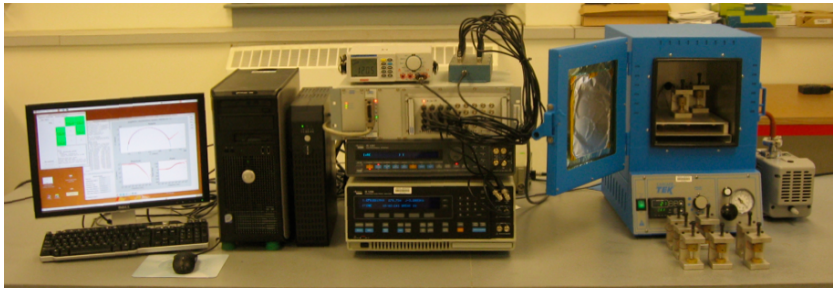


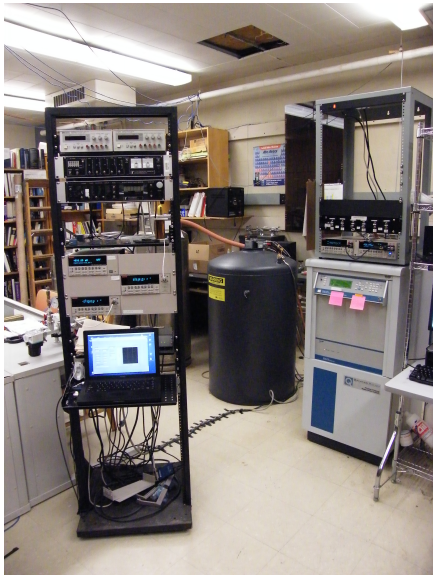
Experimental Work

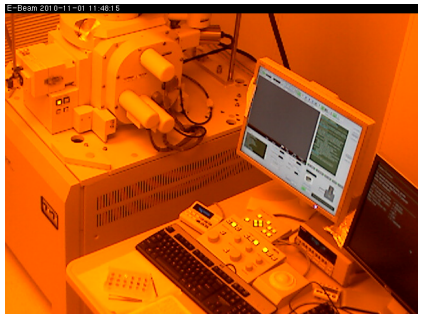
Our group specializes in the fabrication and characterization of materials and nanoscale systems with novel physical properties.

- many experiments require control of multiple instruments, both commercial and custom
- temperature-controlled impedance measurements, for frequency dependent charge transport characterization
- low temperature electronic and magnetic properties measurement systems
- computerized microscopy and fabrication systems, AFM, SEM, e-beam writing

Lab Objectives





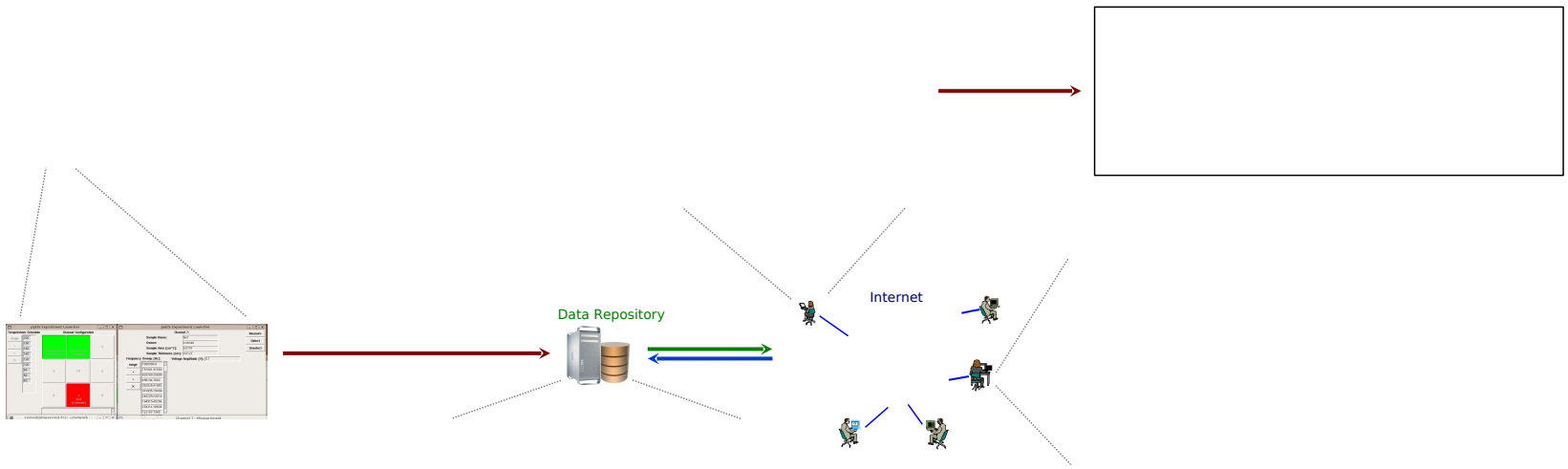


Informatics Solutions

We strive to implement *informatics* techniques to make our research more efficient

- rapidly explore system design parameters
- ensure traceability of experimental details
- enhance feedback to collaborators
- free up our time for more creative tasks

Informatics Capabilities



Development resources for small research groups are limited, so we identified a minimal feature set for an *informatics-enabled system*:

- automation of data acquisition for multiple samples in parallel
- modular hardware design, which is easy to maintain, replace, adapt, and reconfigure for new needs
- tracking of important experimental details, metadata
- scalable organization of large volumes of data
- computer-assisted data analysis and visualization
- ability to work remotely across the Internet

Implementation Obstacles

Condensed matter physics and materials science labs like our own often contain numerous different experiment stations composed both of commercial "turn-key" solutions and custom setups built from modular hardware components

- legacy equipment with proprietary and out-dated interface technology creates compatibility problems and IT maintenance nightmares
- *vendor lock-in* makes updating and migrating computer systems costly and painful
- heterogeneity and incompatibility of data formats creates bottlenecks in the research work-flow
- restrictive software licenses limit number of "work stations" for analysis and simulation work
- large volumes of data become increasingly difficult to handle using manual "spreadsheet" analysis techniques

Development Methodology

Programming Platform Choice

We propose that small research groups can leverage the power of open source software and high-level "scripting" languages (e.g., Python, Tcl/Tk, Perl, or Ruby) to knit together existing tools, each of which excel at various parts of the complete informatics package

- contrasted to compiled languages (e.g., C/C++, Fortran, Java), there are benefits *and* drawbacks depending on the application
- rapid-development capability enabled by simple flexible syntax and elimination of complex compilation steps
- languages designed to optimize the developers' use of time, not the computer's
- powerful built-in tools for both textual and binary information handling
- availability of high-quality open source extensions for array-based numerical analysis, plotting/graphics, graphical user interfaces, web-application development, and much more
- permissive "open source" licensing which encourages sharing of code and interoperability
- cross-platform capability for applications that run on Linux, Mac OS X, and Windows

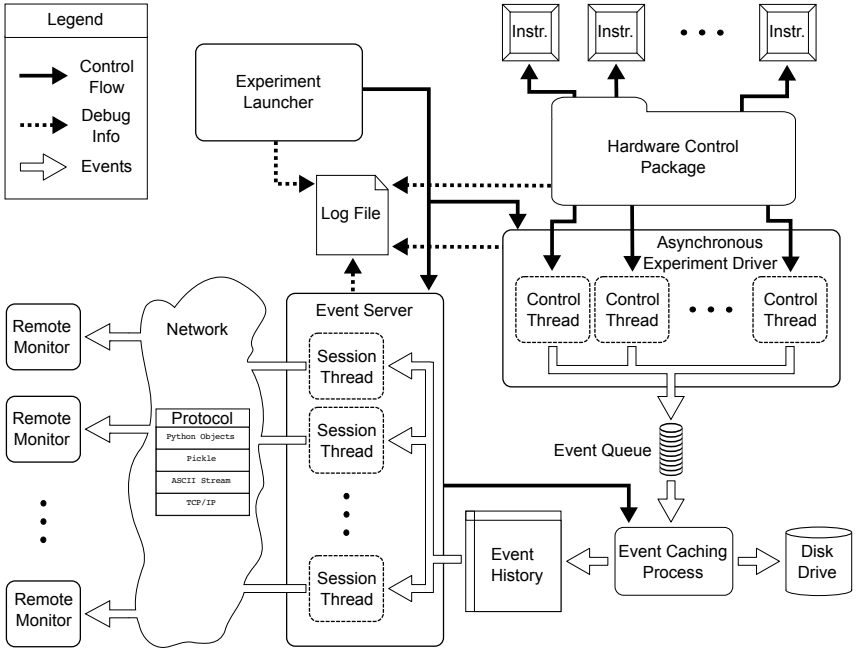
Development Methodology

Framework Building

During the course of developing a particular application many software components must be developed that could be useful for future applications

- organize code base into a hierarchy of reusable modules of functionality ("libraries" and "packages")
- eliminate redundancy ("copy and paste" coding) to minimize bug-propagation and maintenance efforts... "Don't Repeat Yourself"
- rapidly construct new features by combining existing implementations
- encourages distribution potential and code sharing through generality and extensibility
- modern scripting languages provide simple development tools which make framework building feasible for non-professionals

Example Applications

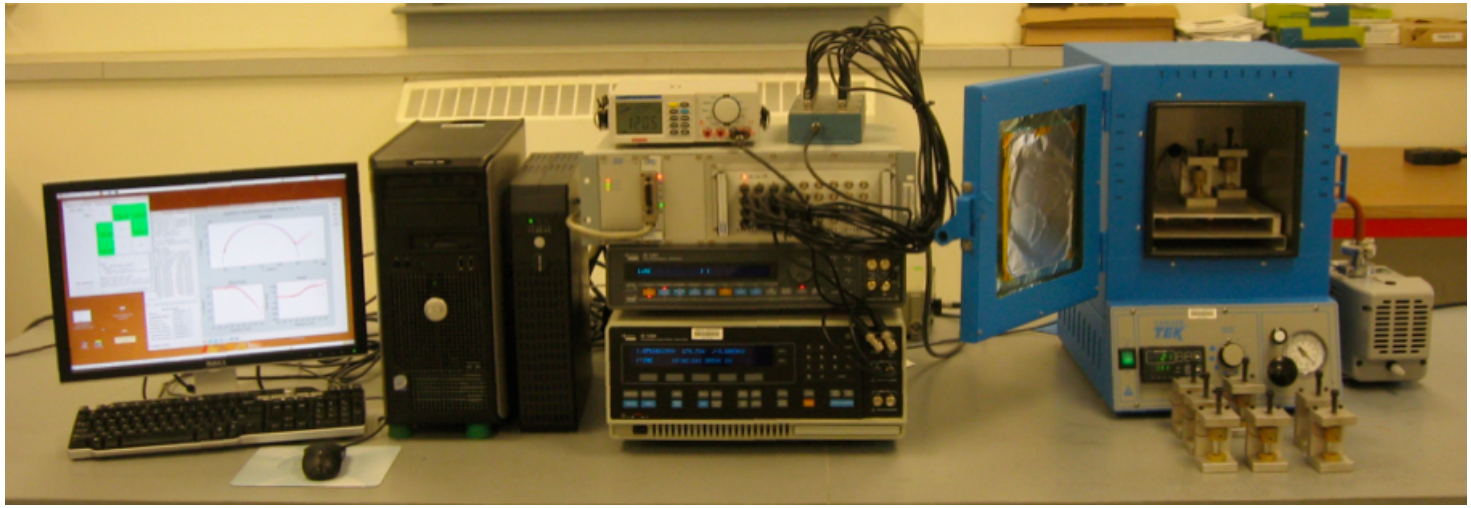


Automat - Experimental Automation and Informatics Framework/Toolkit

Using the open source Python scripting language we developed a pilot "middleware" framework

- serves as foundation and reusable code-base for more specialized user applications
- provides application programming interfaces (APIs), but not user interfaces
- simplifies the challenges of multiple asynchronous device control

Example Applications

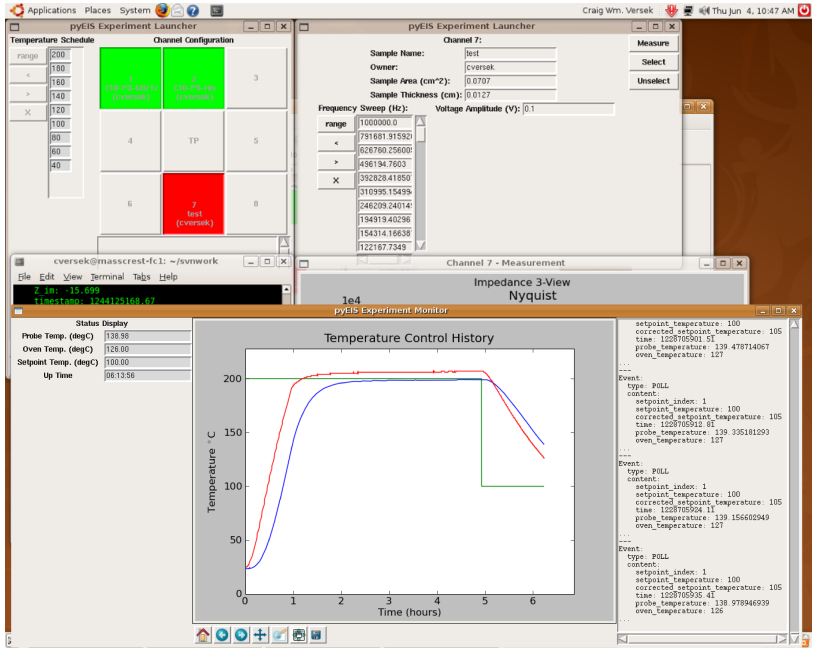


pyEIS - Informatics Tools for Impedance Spectroscopy

Building on top of the *Automat* framework, we developed a custom suite of software applications to give enhanced informatics capabilities to a new customized equipment setup

- developed in Python on Ubuntu Linux, but portable to other platforms
- modular software and hardware design philosophy
- 8 multiplexed sample channels, temperature control, vacuum or humidified air environment
- impedance and other electrical measurements

Example Applications

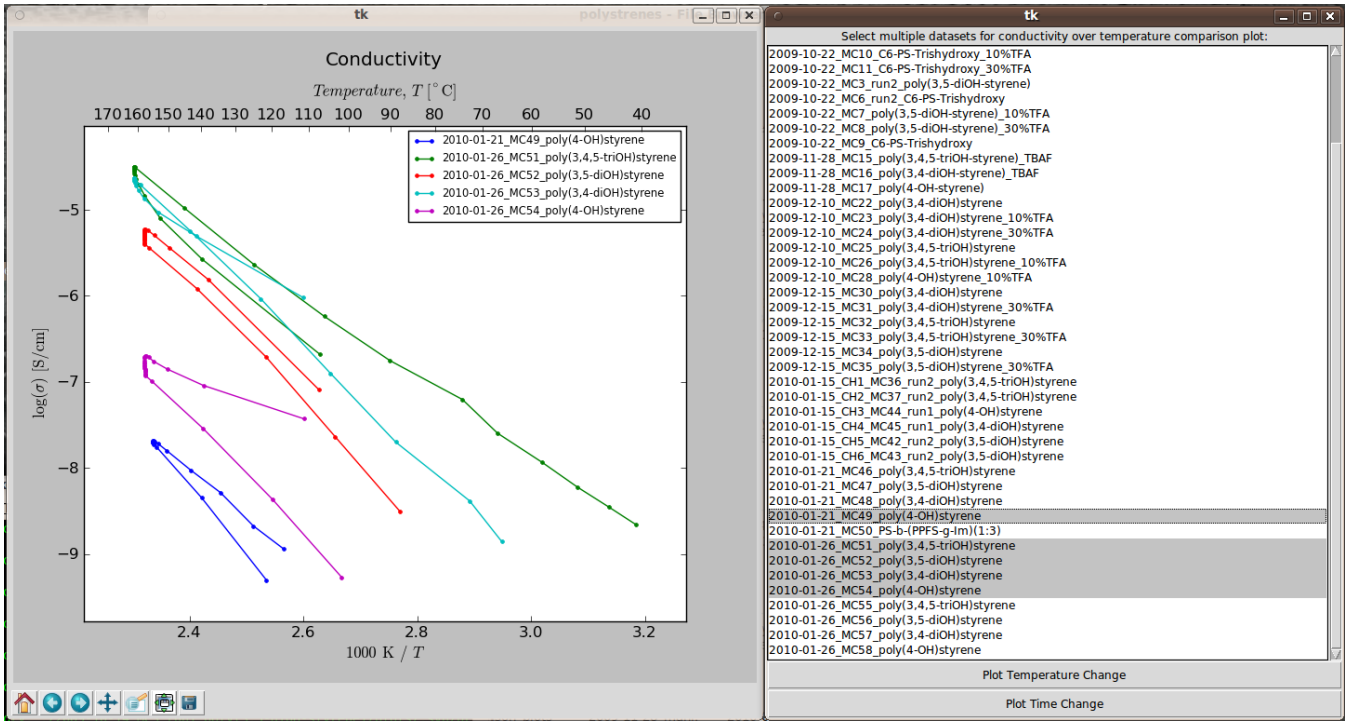


pyEIS - Experiment Launcher

User application for configuring and running experiments

- simple graphical user interface
- extensive metadata collection and preservation
- remote monitoring of the experiment state
- on-the-fly data file caching into organized file structures
- data exported in *Zplot* compatible format

Example Applications



pyEIS - Impedance/Conductivity Analysis Tools

User applications for rapidly analyzing large volumes of data

- quick, interactive curve fitting to reduce data by the batch
- rapid plotting and comparison of samples, using data-discovery tools
- data can be exported to spreadsheet formats

Conclusions

- heterogeneous software and hardware interfaces can lead to fragmented work-flow
- custom programming can help bridge gaps in the flow of experimental information in small research group settings
- scripting languages are an effective tool for rapid informatics capability development